**MEDICAL INVENTORY MANAGEMENT**

COLLEGE NAME: Tiruppur Kumaran College For Women

COLLEGE CODE: bru3j

**TEAM ID**: NM2025TMID28417

**TEAM MEMBERS:**

**Team LeaderName :** ANANDHI N

**Email** : anandhinarayanan06@gmail.com

**Team Member 1**  **:**  DEVIKA

**Email :** devikarajpurohit027@gmail.com

**Team Member 2 :**  THANGAMANI R

**Email :** tt829891@gmail.com

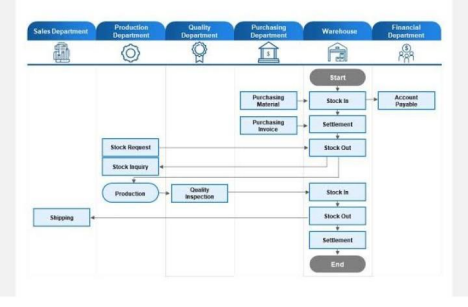
**Team Member** **3**  **:**  SUKAINA S

**Email :** sahilabanu748@gmail.com

**1.INTRODUCTION**

**1.1 Project overview**

Medical inventory management is the process of planning,organizing and consumables , medical equipment in healthcare organizations. It ensures that the right product is available at the right place,in the right quantity and at the right time.

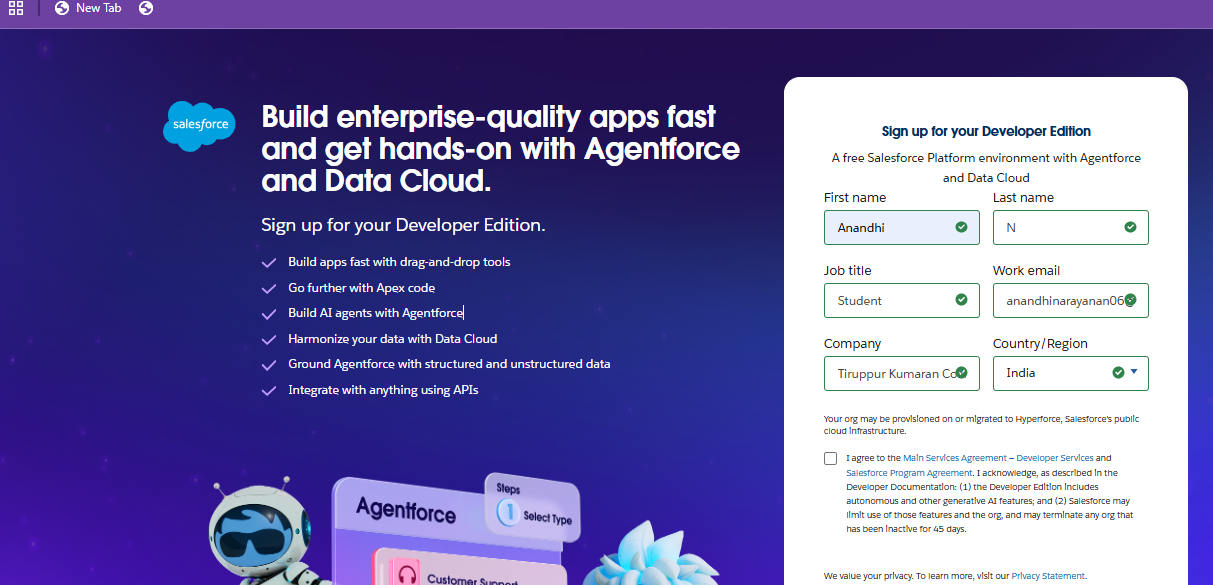
**1.2 Purpose**

* To maintain uninterrupted supply of essential medicines &equipment
* To minimize wastage of resource due to expiry or damage
* To improve operational efficiency in hospitals and clinics
* To ensure cost-effective procedurement and usage
* To support better patient care services

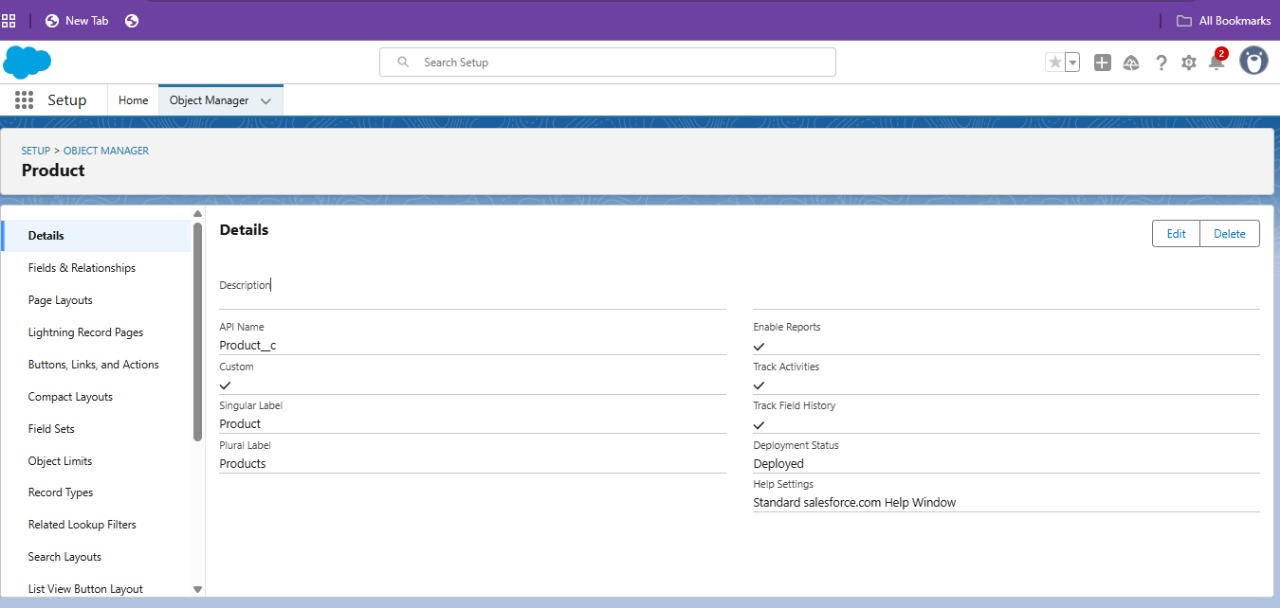
**DEVELOPMENT PHASE**

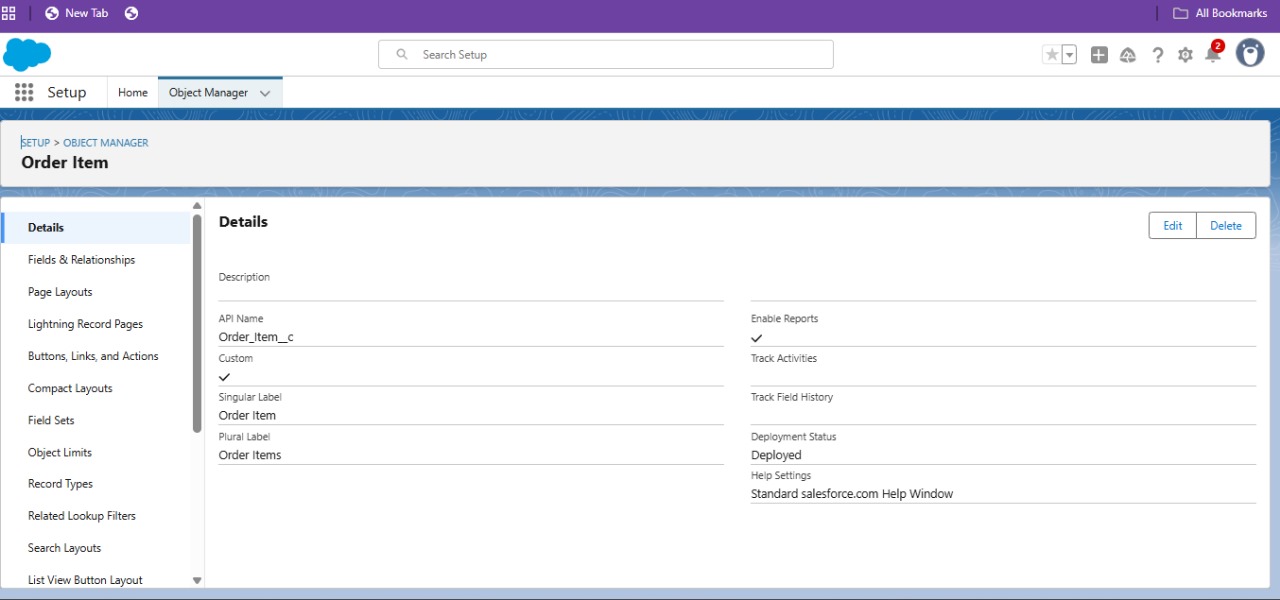
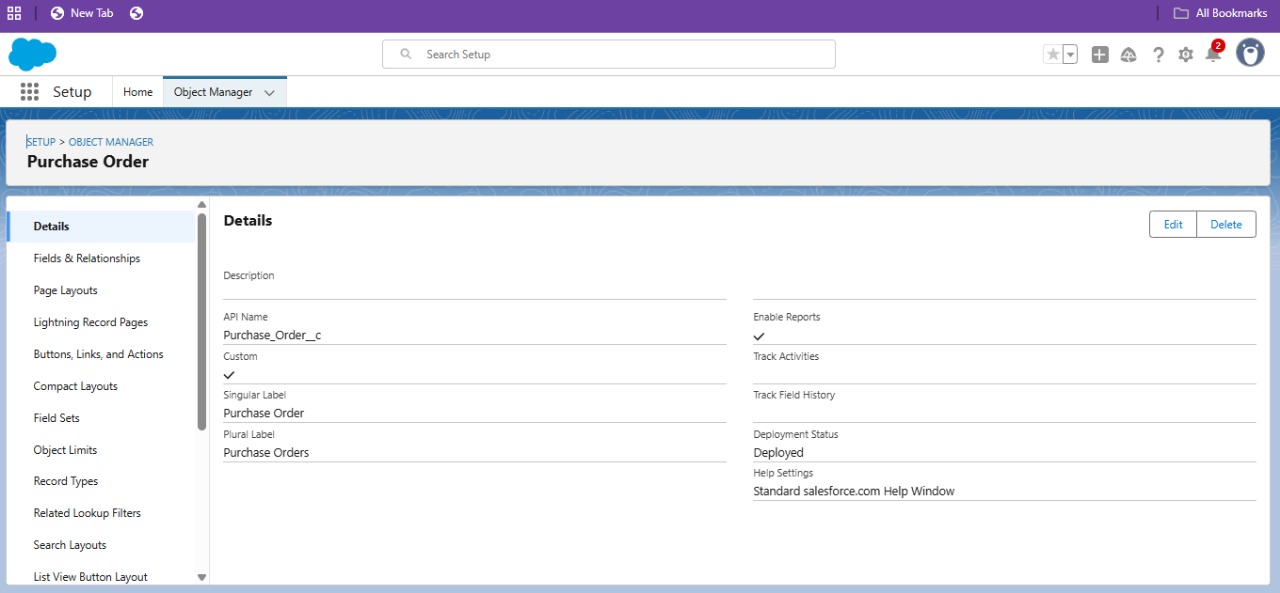
**Creating Developer Account:**

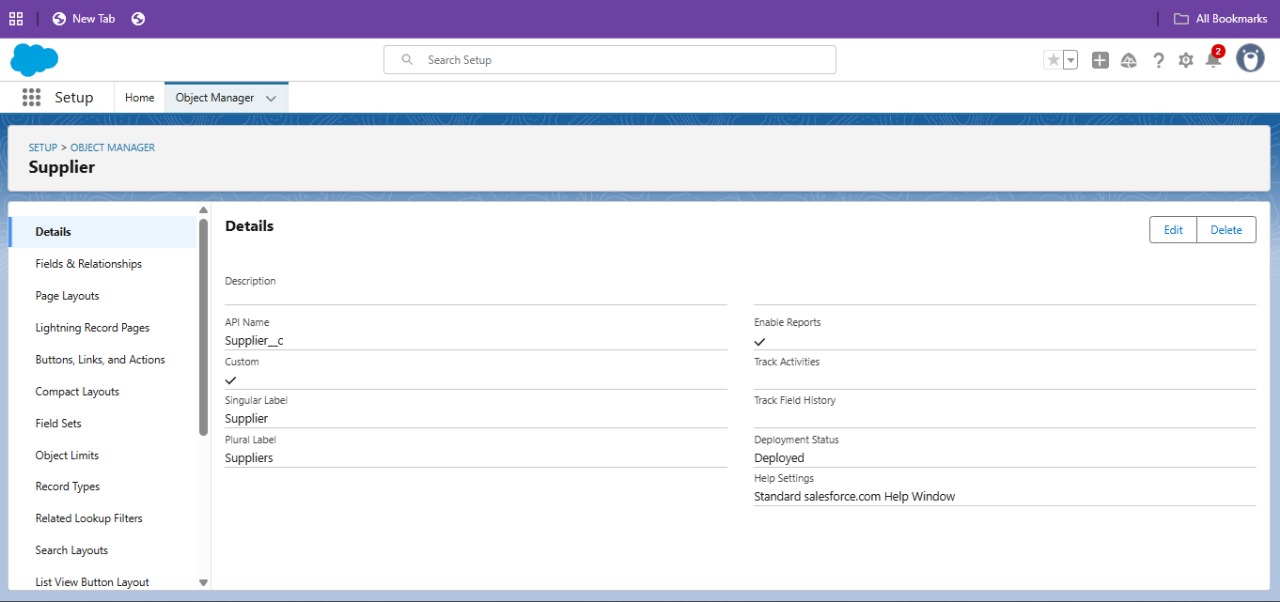
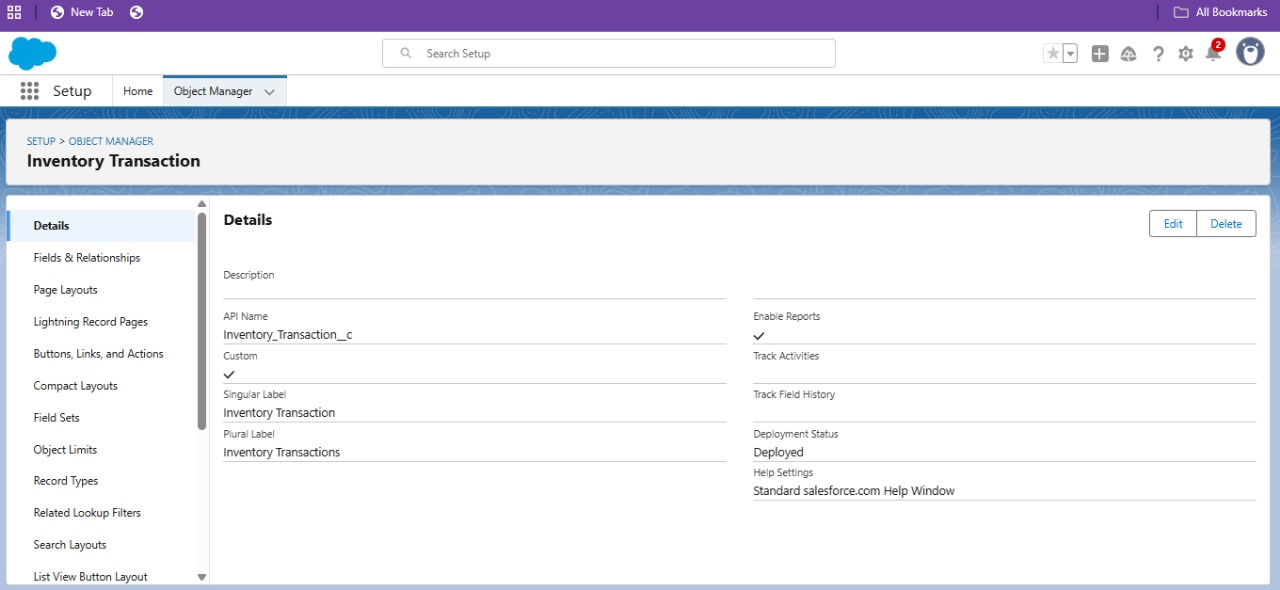
By using this URL – <https://developer.salesforce.com/signup>



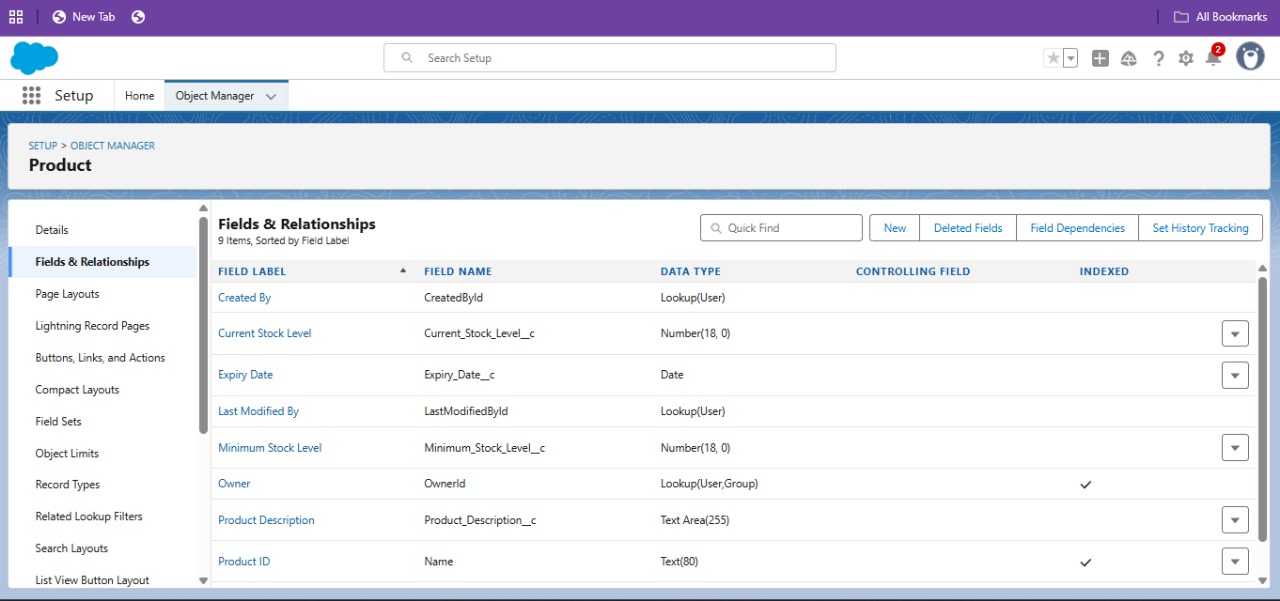
* Created object: Product, Purchase Order, Order Item, Inventory Transaction and Supplier objects.

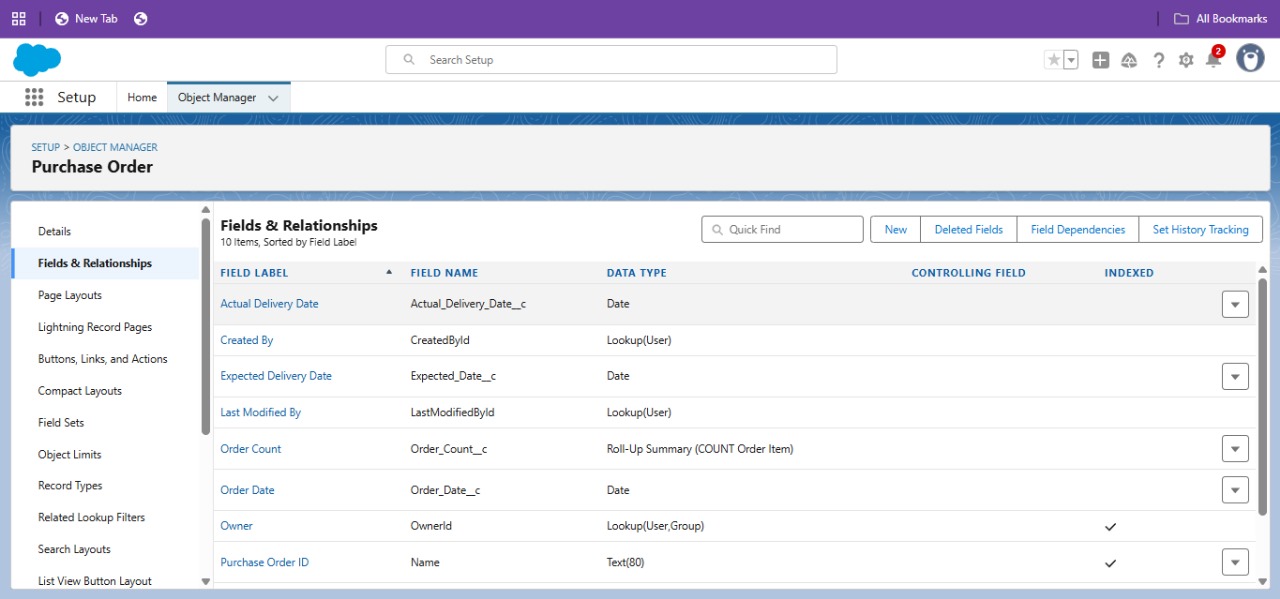


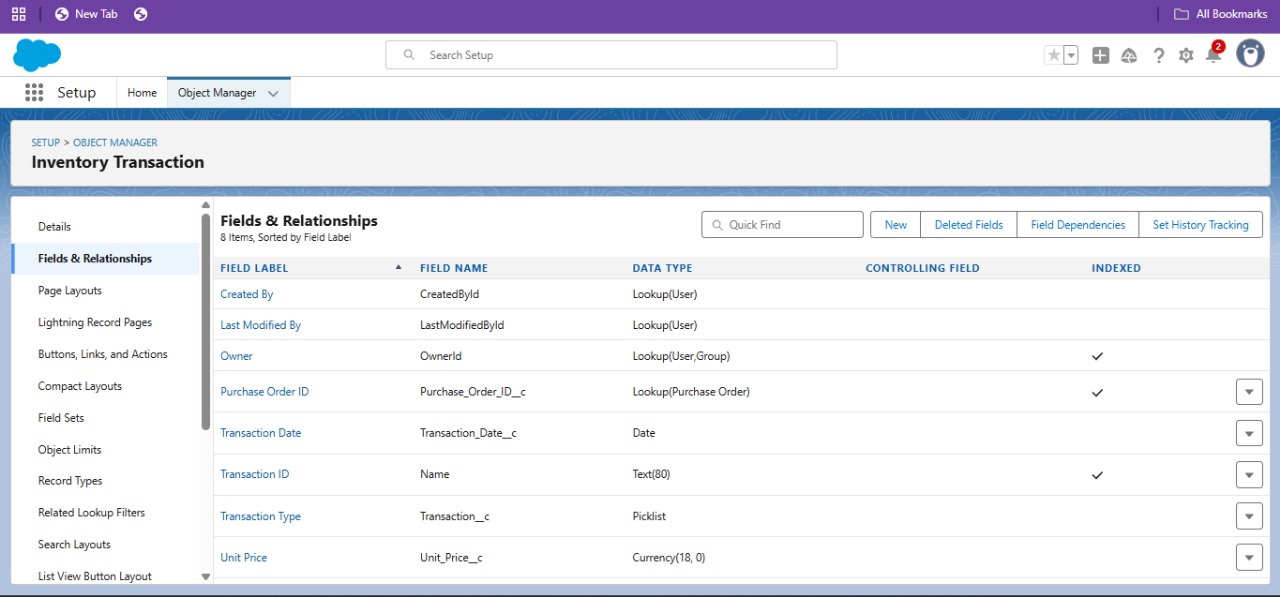
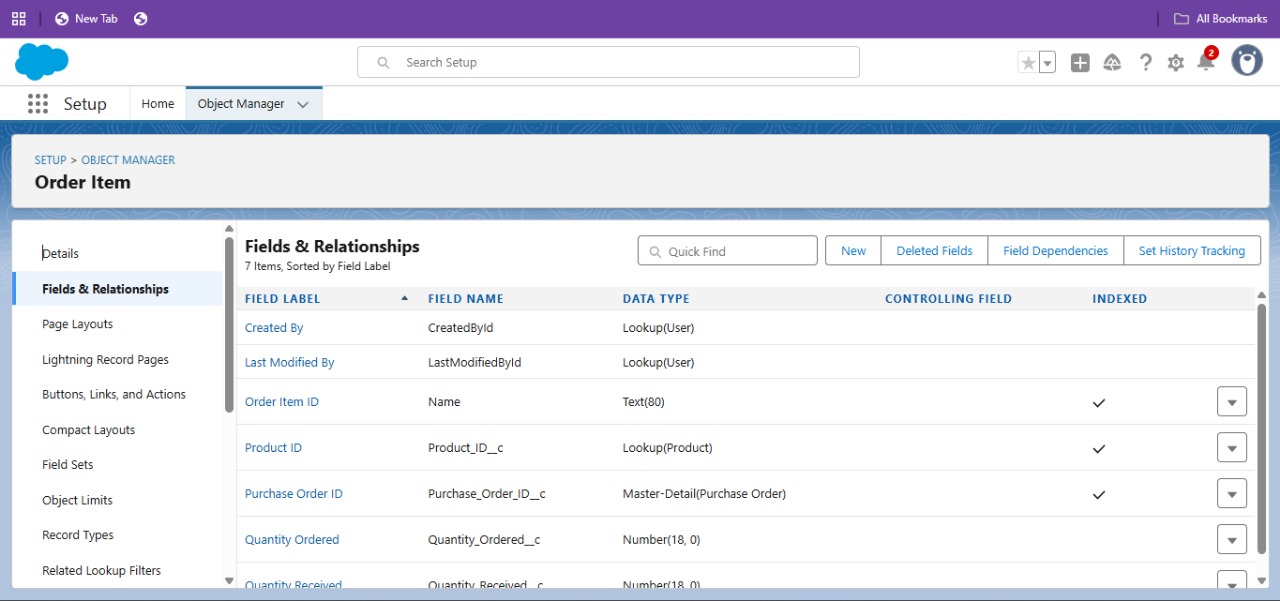


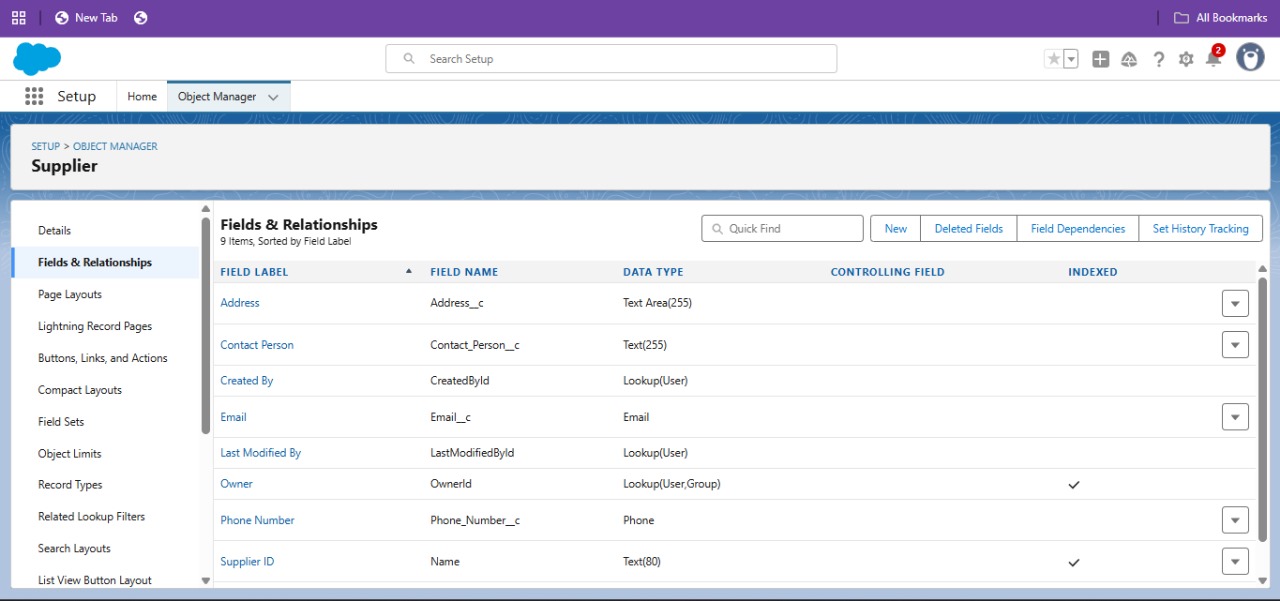


* Configured fields and relationships

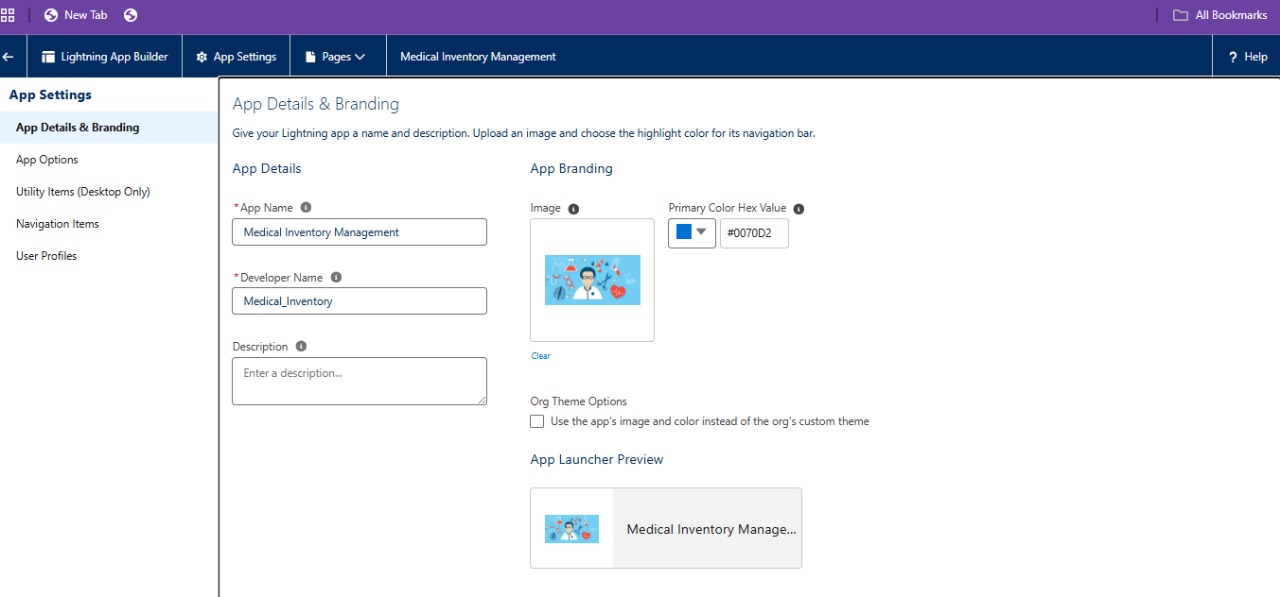


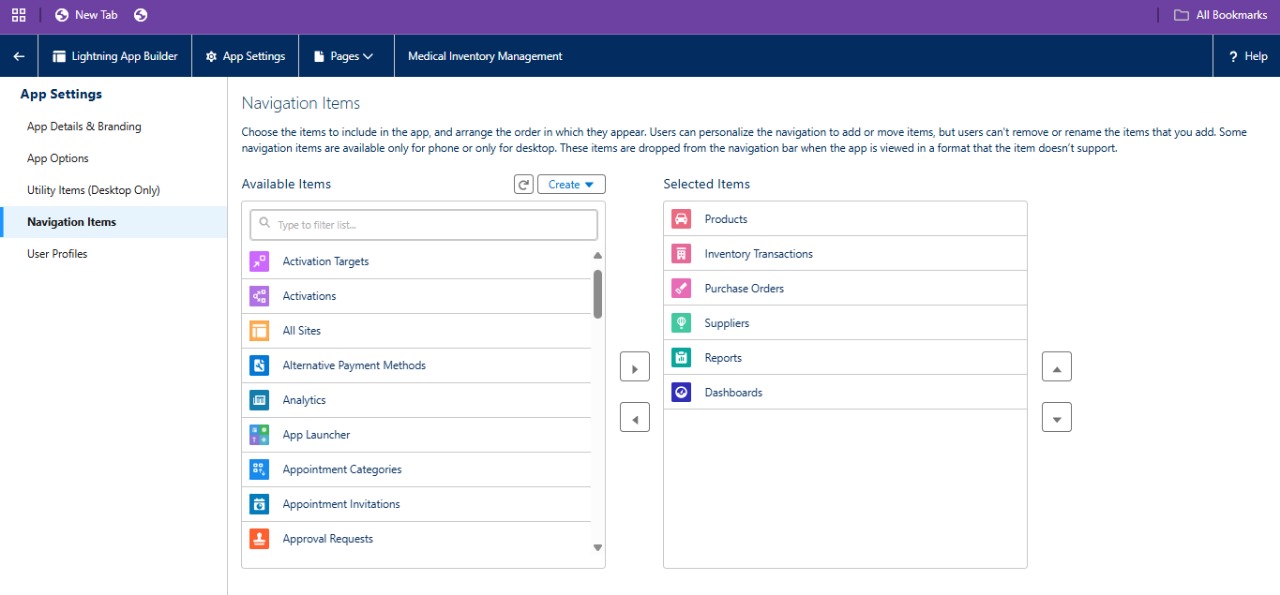


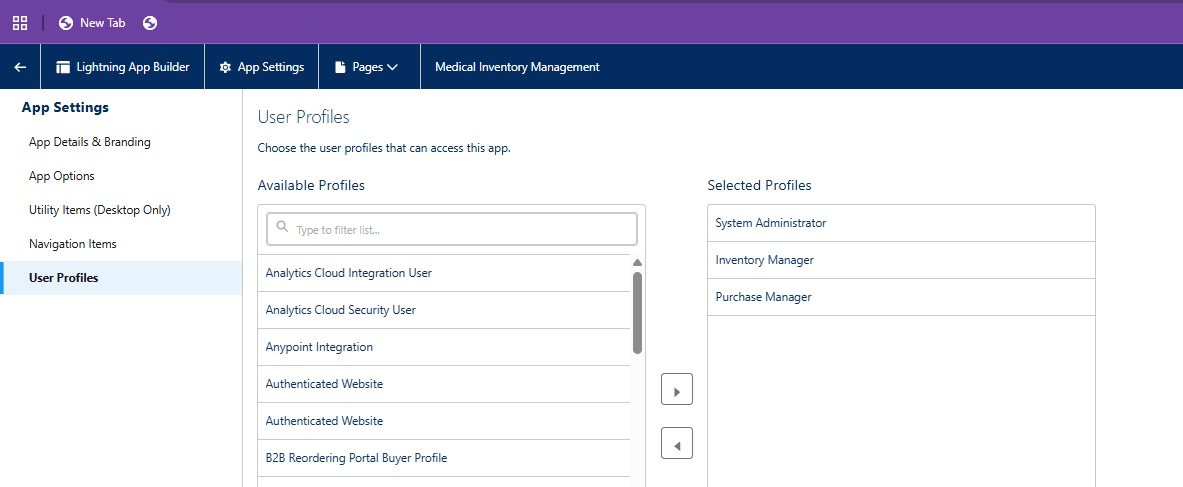




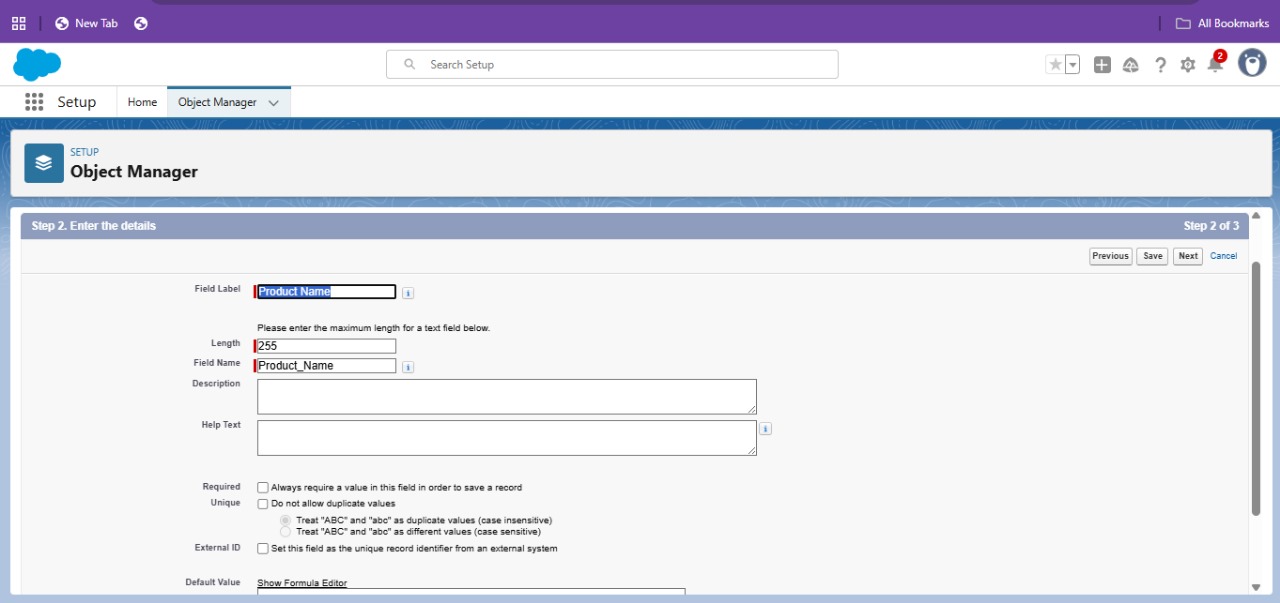
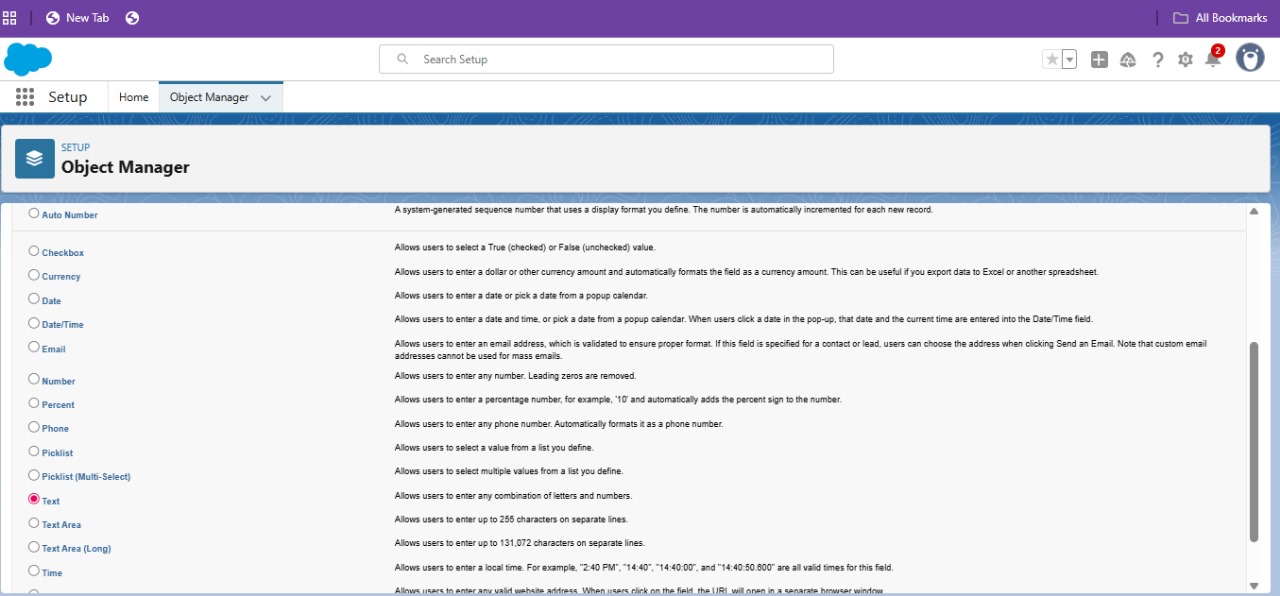
* Developed Lightning App with relevant tabs

∙

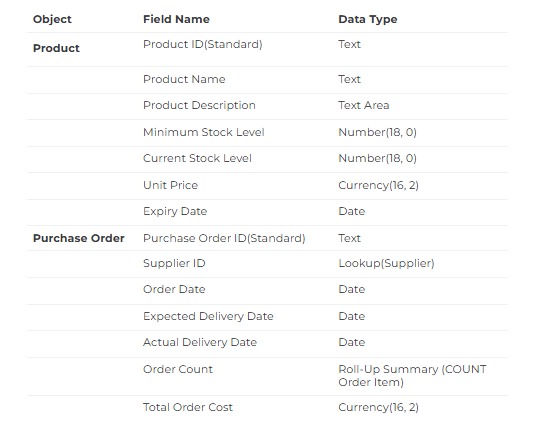
∙

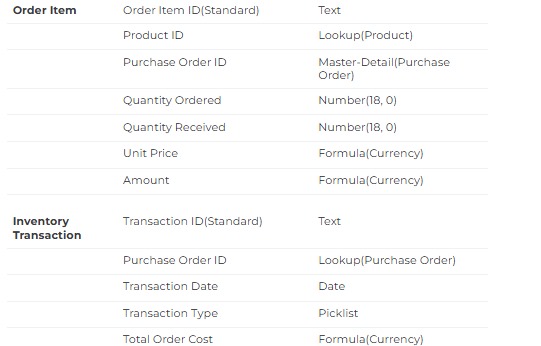


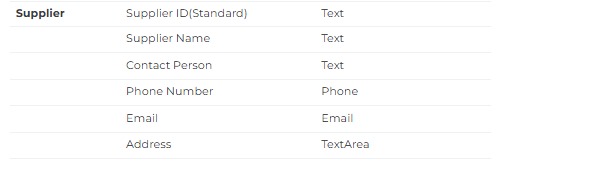
* Now, add every field with its desired datatypes.



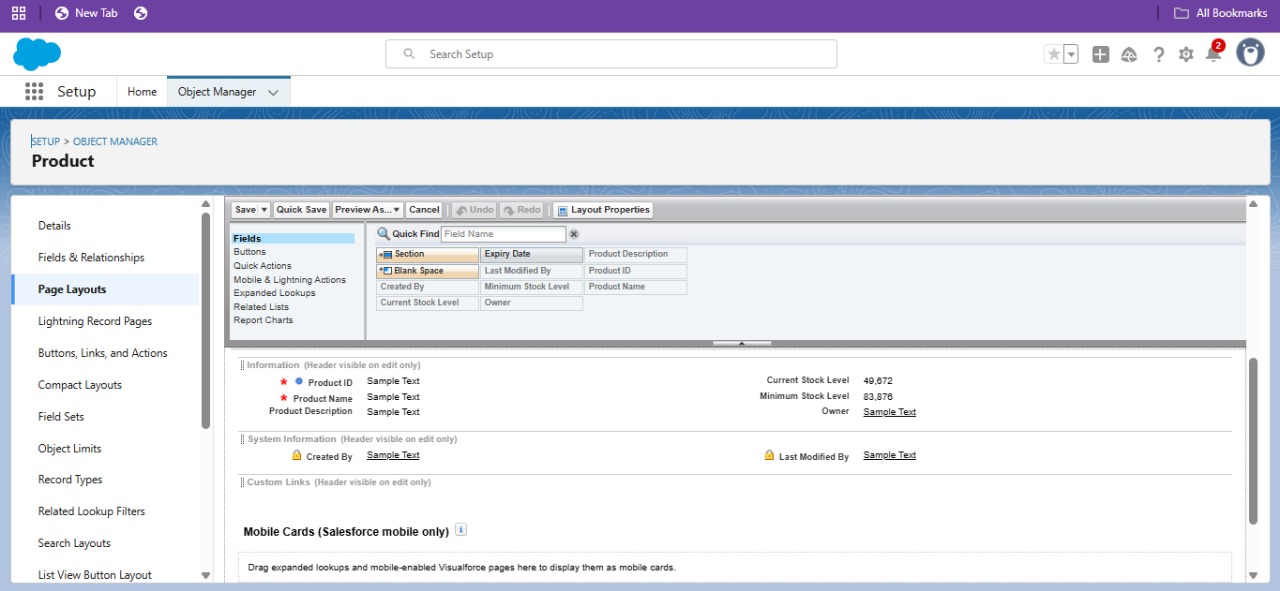
* Then add every objects with its field name and datatypes



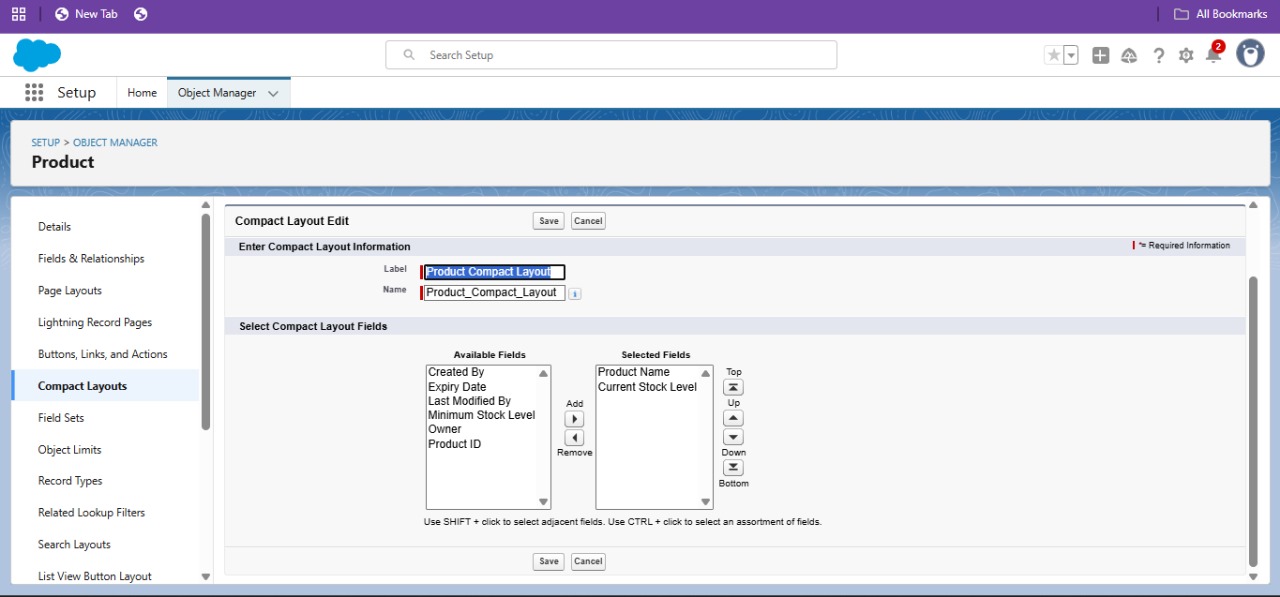




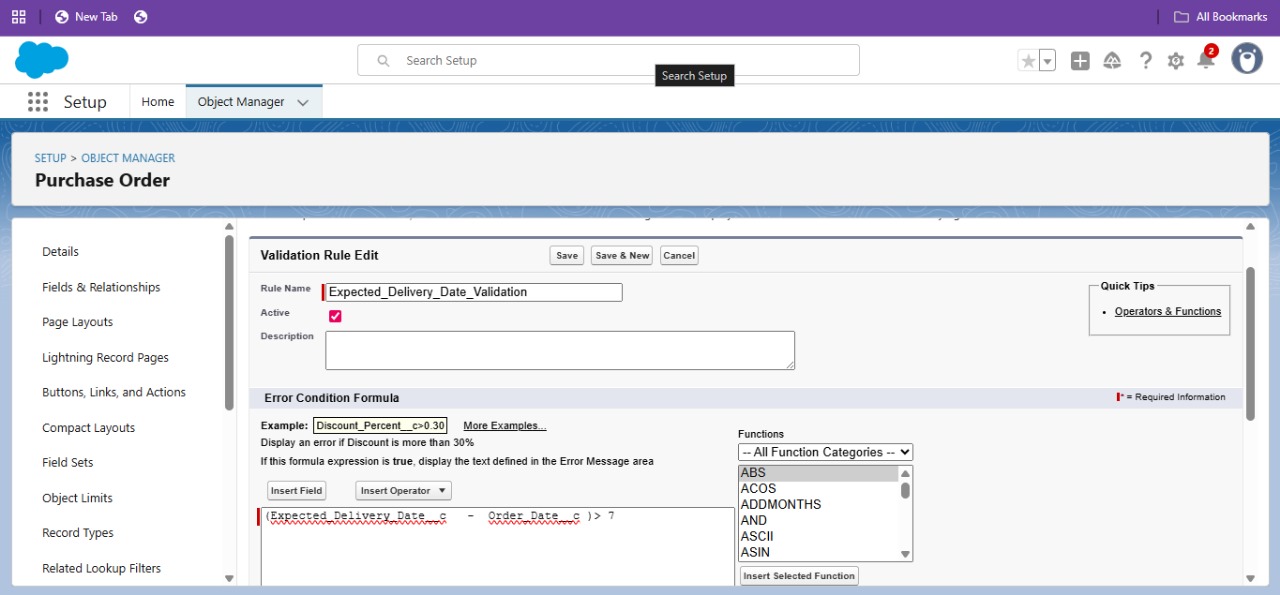
* Next edit a Page Layout in Product Object in every fields like product, purchase order, order item, inventory transaction, supplier

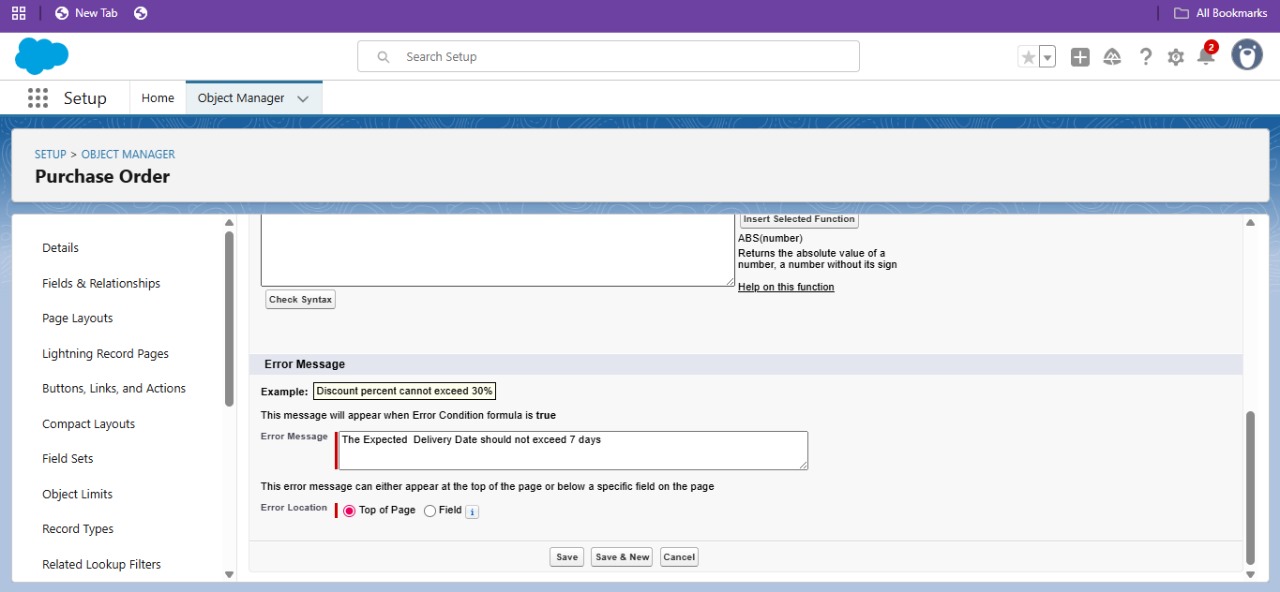


* Next create Compact Layout to a Product Object for product and purchase order

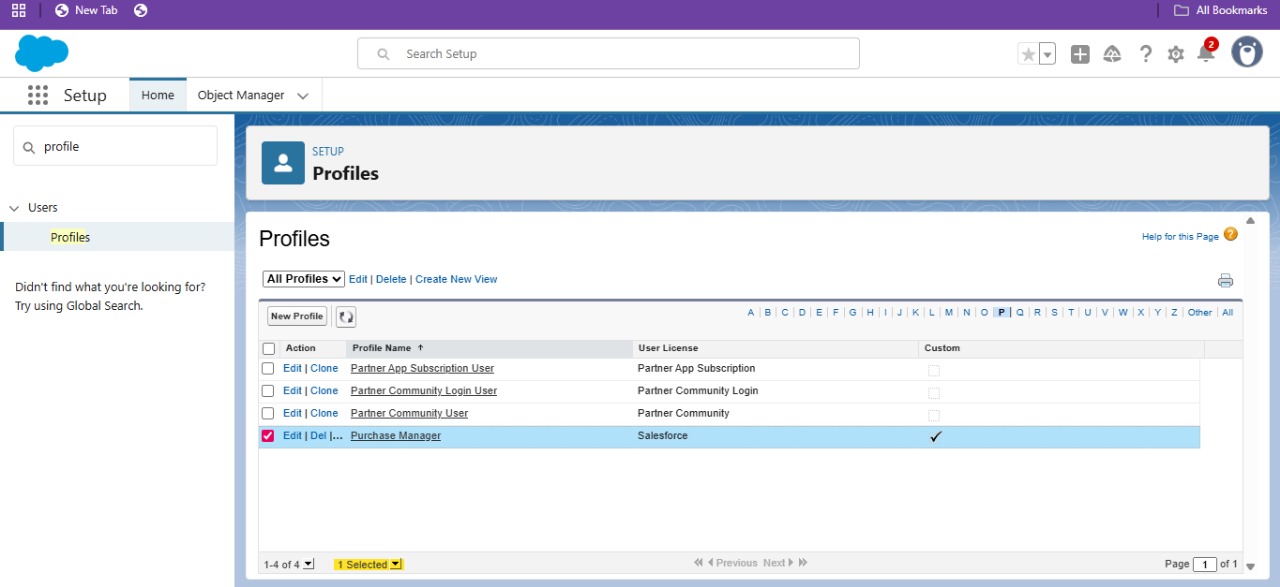


* Then create an Expected Delivery Date Validation rule to a Employee Object in purchase order.

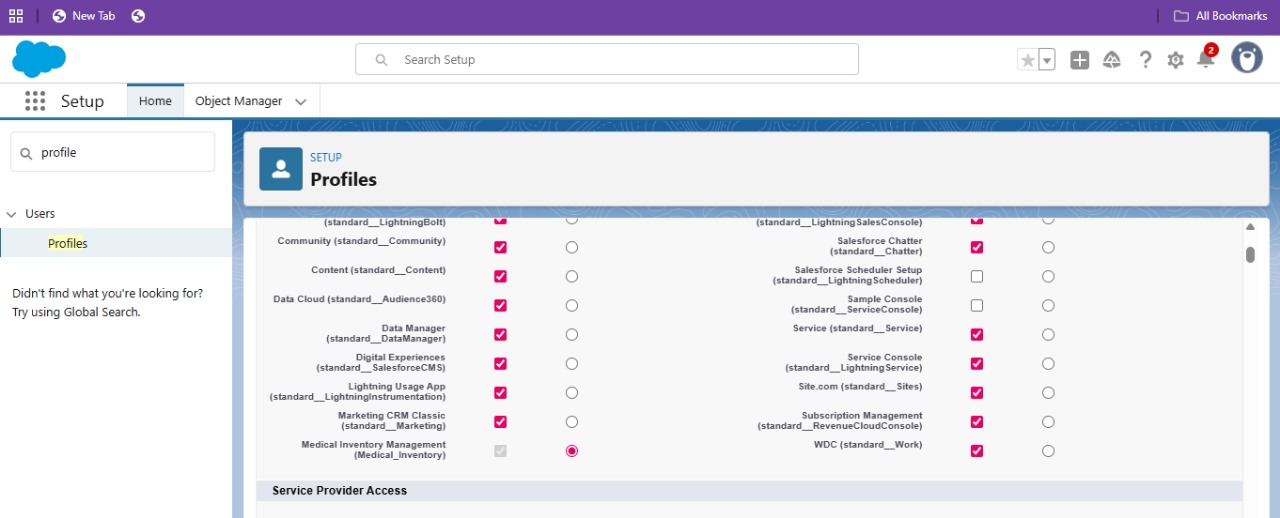




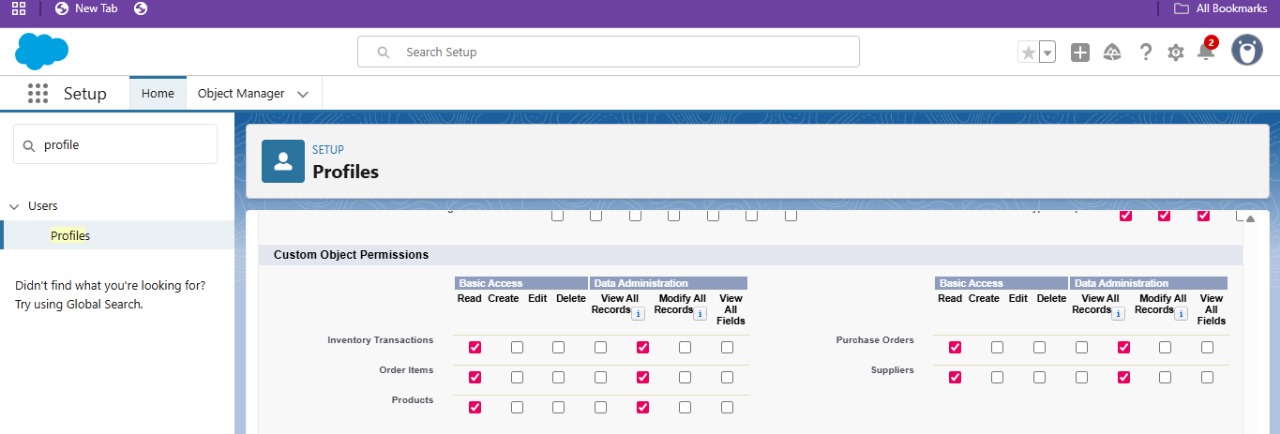
* Next step is to create an Inventory Manager Profile and a Purchase Manager Profile.

∙

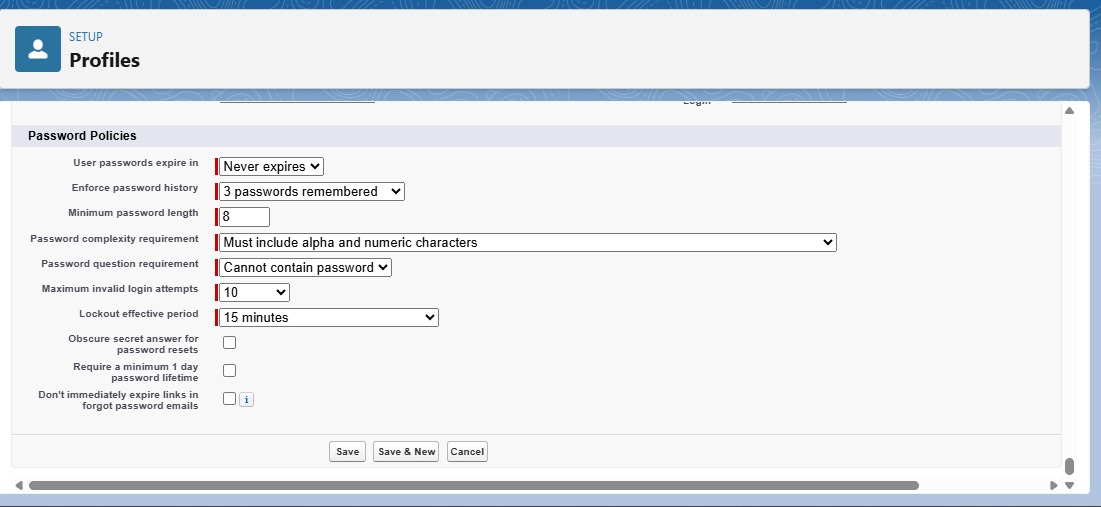
* Select default option for medical inventory management



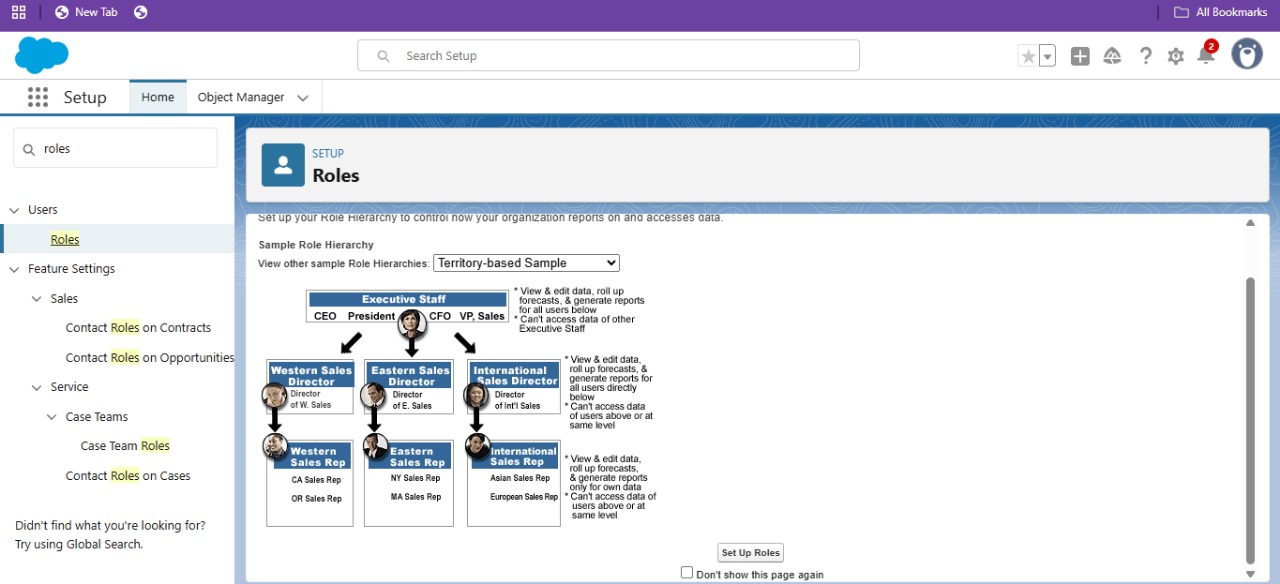
* Then edit Custom Object Permissions as given

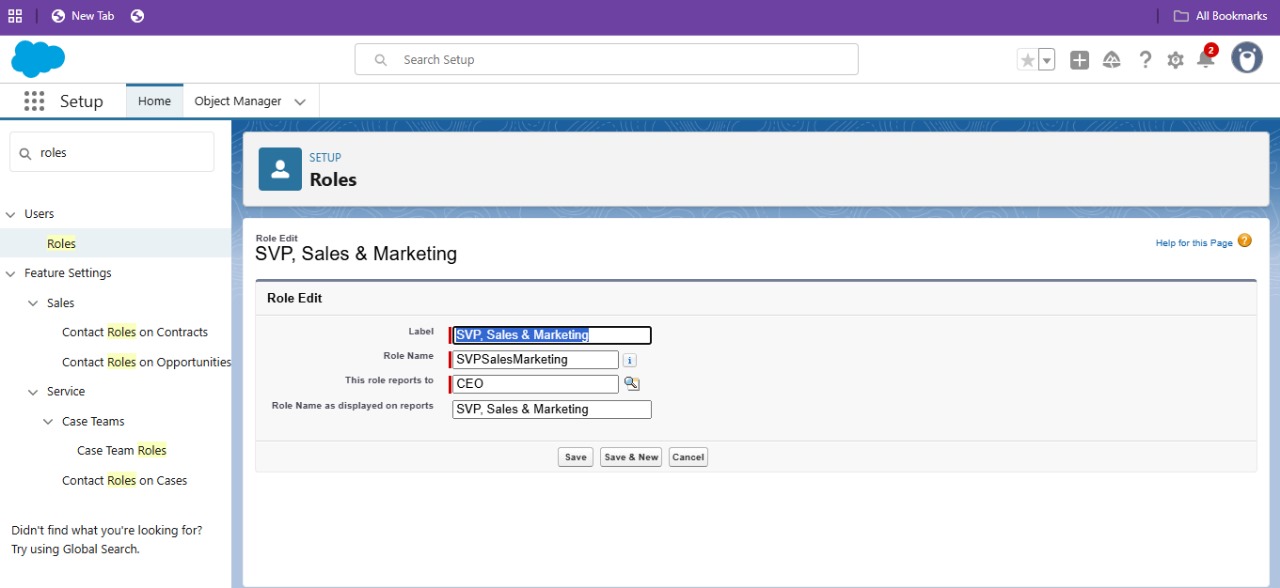
∙ Set

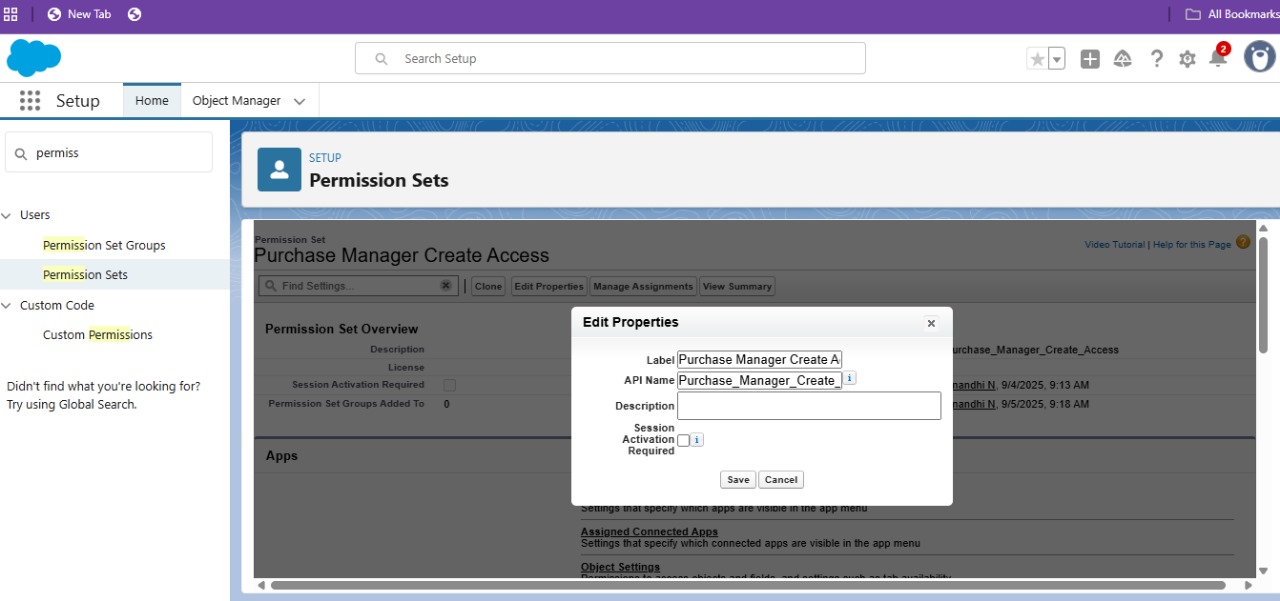
* Password Policies



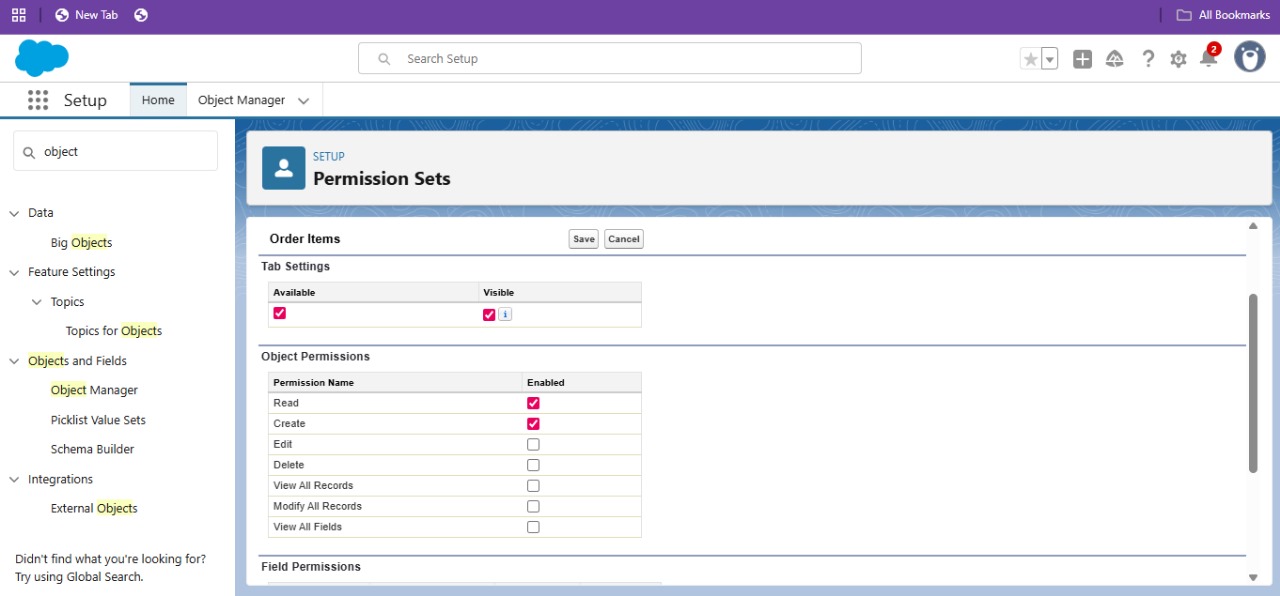
* Next Create a Purchasing Manager Role and Inventory Manager in roles
* Select the setup roles option at bottom



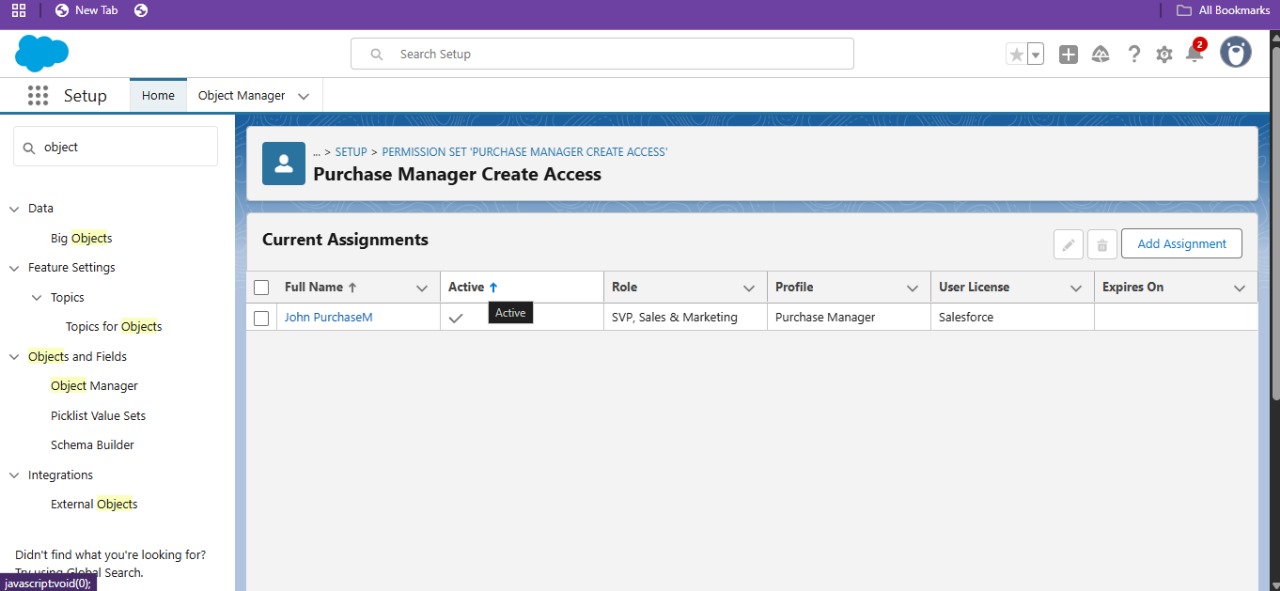
* Then select expand all and click on SVP,sales and marketing. Then fill the label name as purchasing manager and follow the same step to create inventory manager.
* Next step is to create an permission sets for purchase manager



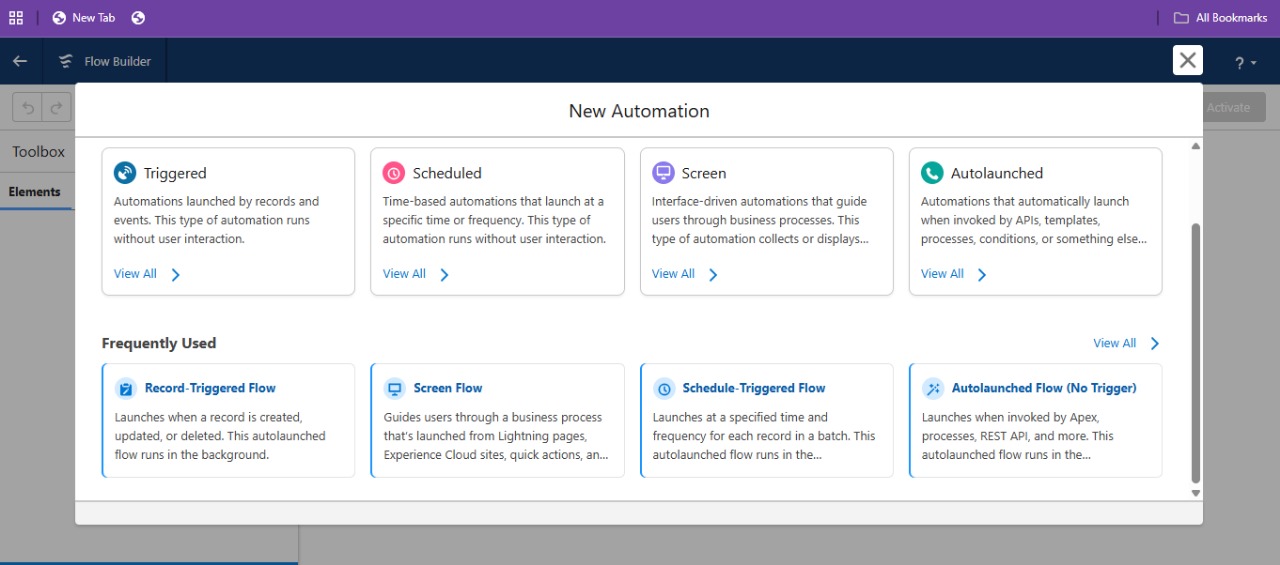
* Then select object settings then select order item. Enable for both Tab Available and Visible.



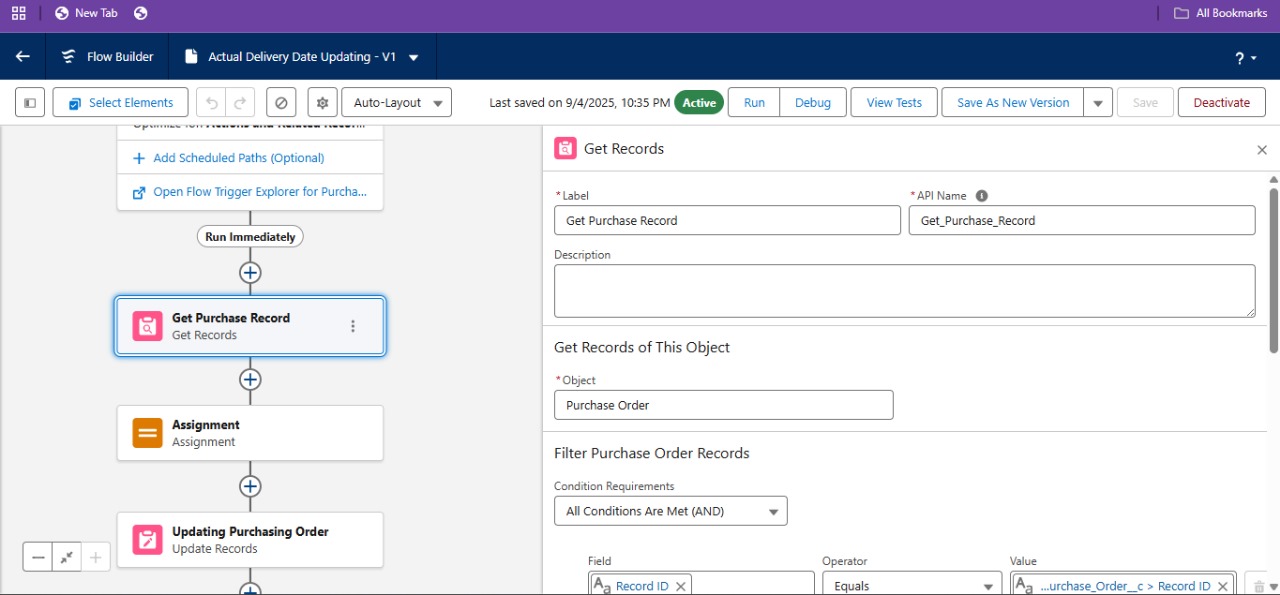
* Navigate to the Permission Set detail page >> Click Manage Assignments >> Click Add Assignments >> Select the user John PurchaseM to assign the permission set to and click Next .

∙

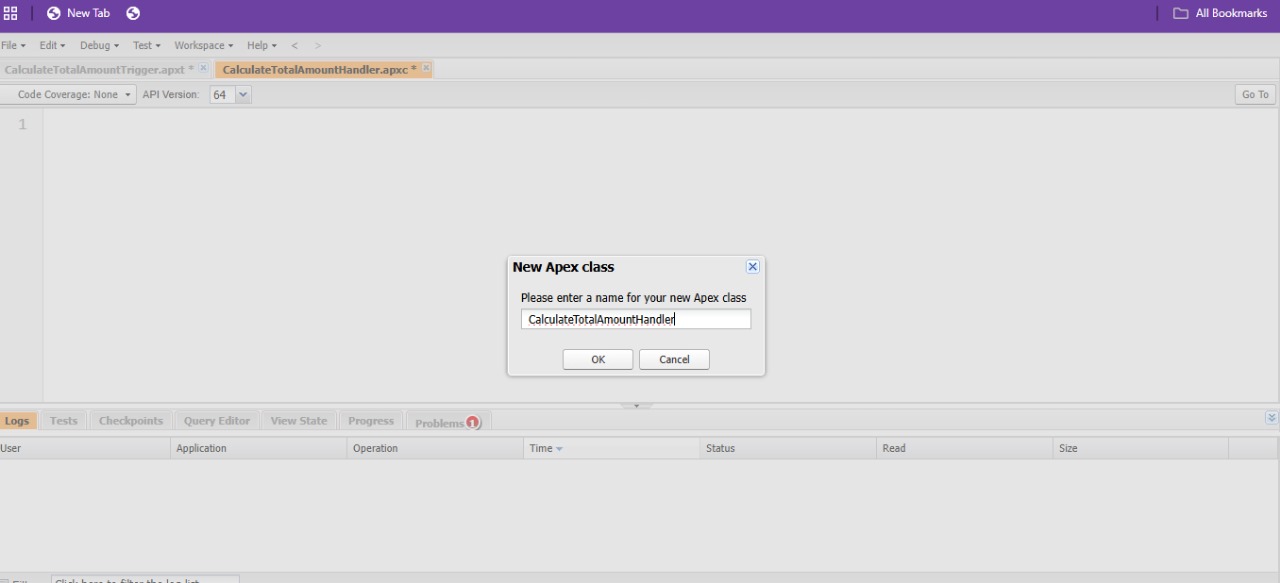
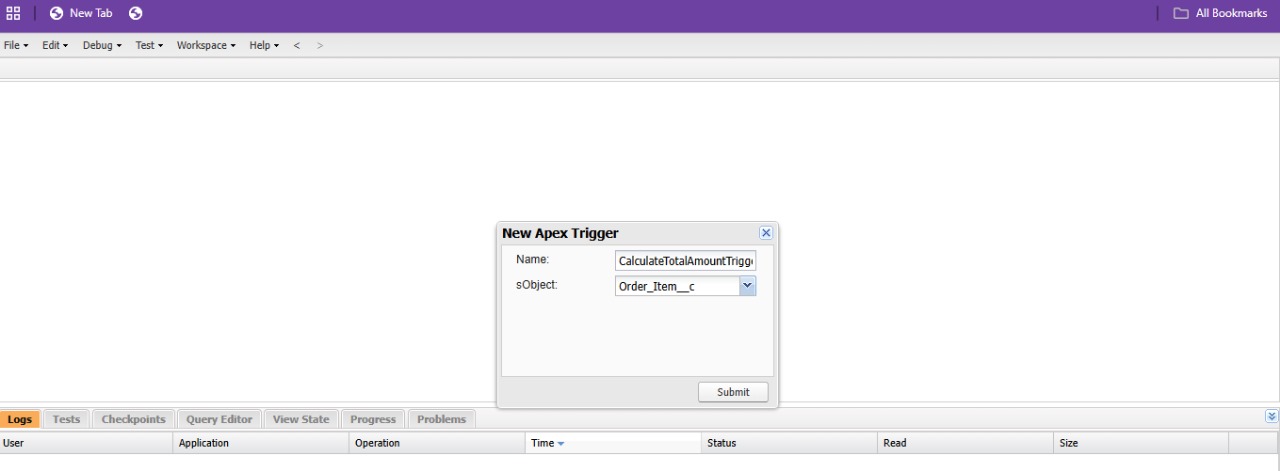
* Then Create Flow to update the Actual Delivery Date and select record-triggered flow.



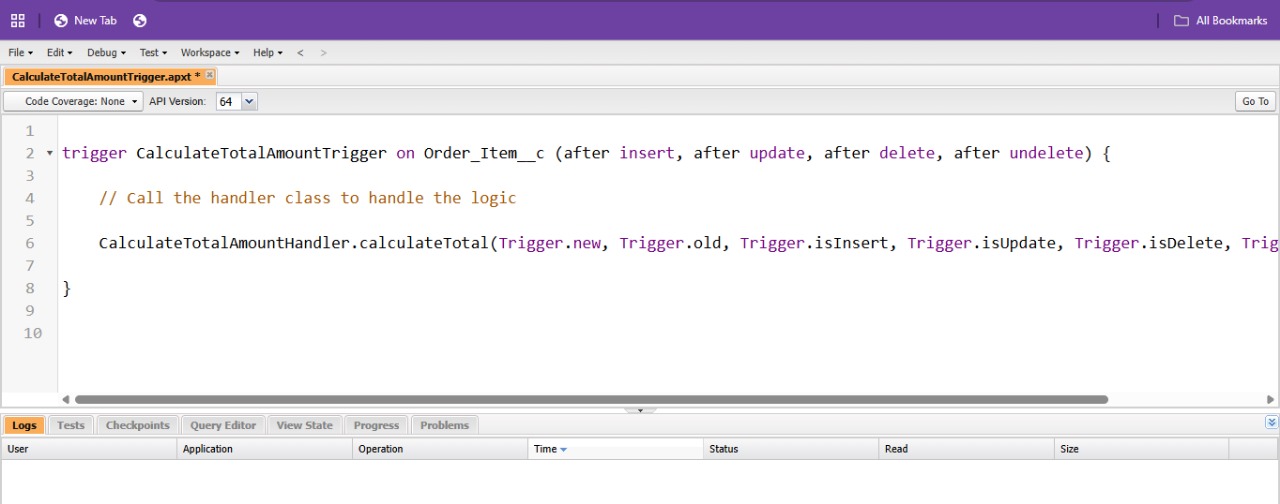
* Under the record trigger flow click on the “+” icon and select Get Records.
* Enter Label as “ Get Purchase Record ”.
* For Object select Purchase Order,
* For Condition Requirements , select All Conditions are Met(AND)Field: Id,Operator: Equal, Value: {!$[Record.Id](http://record.id)}
* Set the field labels and required values in get record.after setting the field values run and click active the final screen should like the picture below.



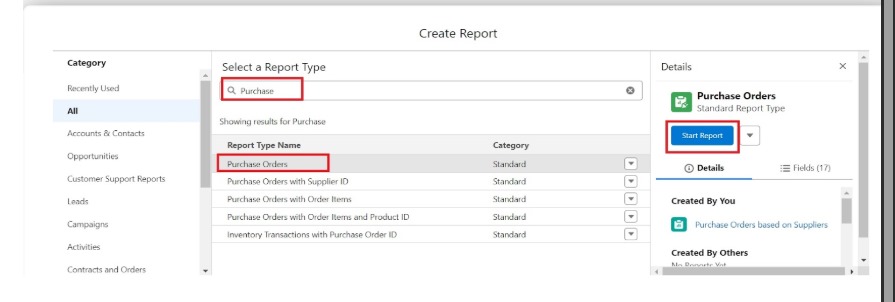
* Then the next step is to Create a Trigger to Calculate total amount on Order Item.
* Click on the Developer Console and create the following Apex trigger and Apex class.

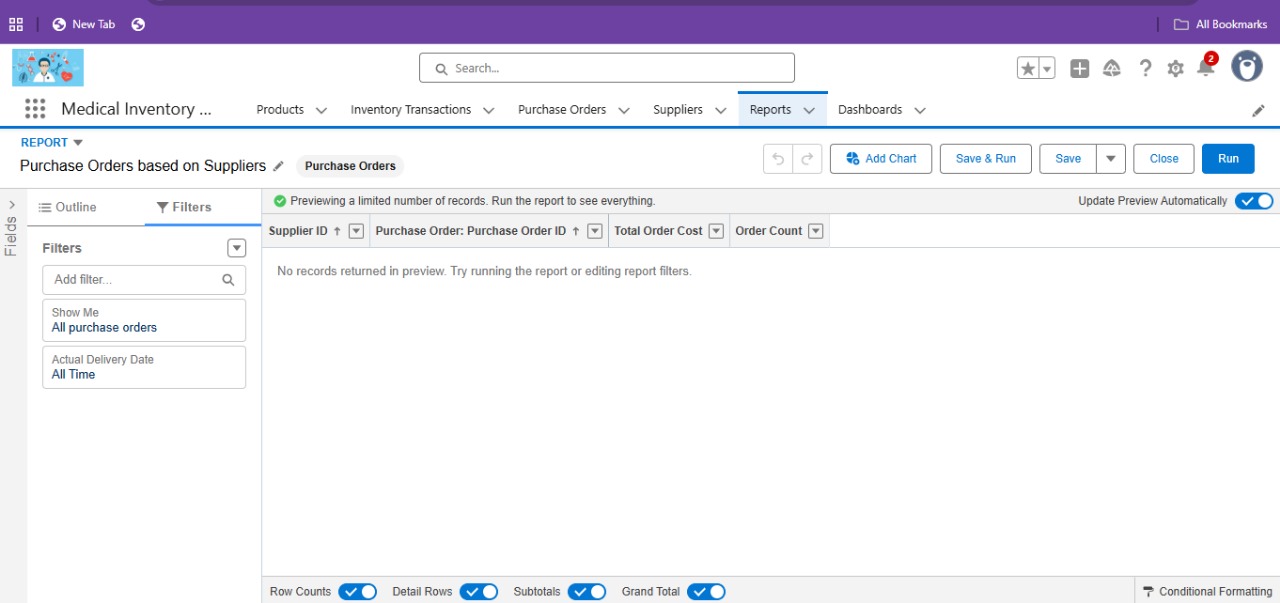


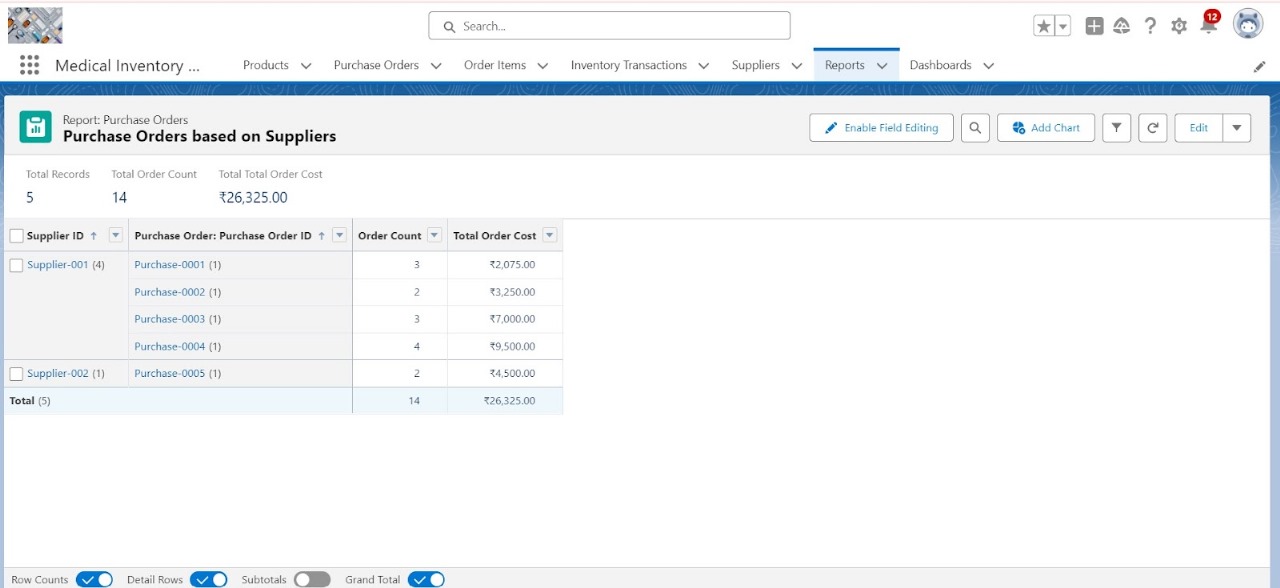
* Then type the coding inside respective apex class and trigger



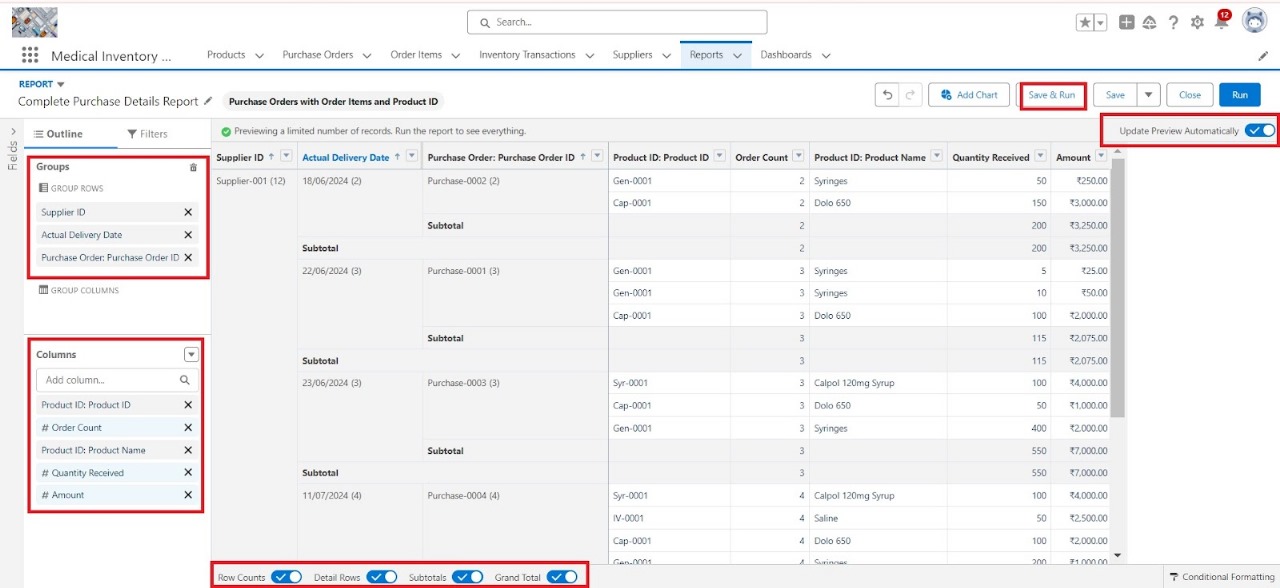
* Next to Create Purchase Orders based on Suppliers(Summary) Report.
* Click App Launcher.
* Select Medical Inventory Management Tab
* Click on Report Tab.
* Click on New Report.
* Click the Report Type as Purchase Orders, click start report.



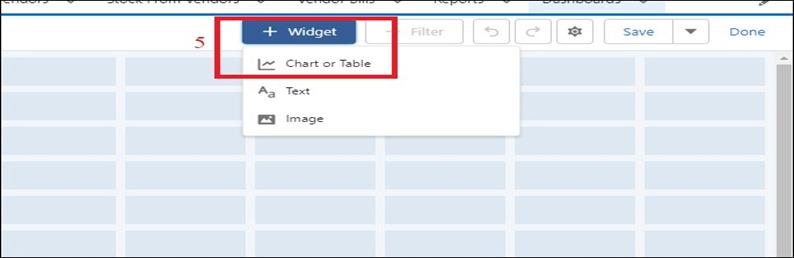
* Click on Filters and select as follows and click on apply.
* Click on App Launcher on the left side of the screen.
* Search Medical Inventory Management App & click on it.
* Click on Reports Tab.
* Click on Purchase Orders based on Suppliers and see records.



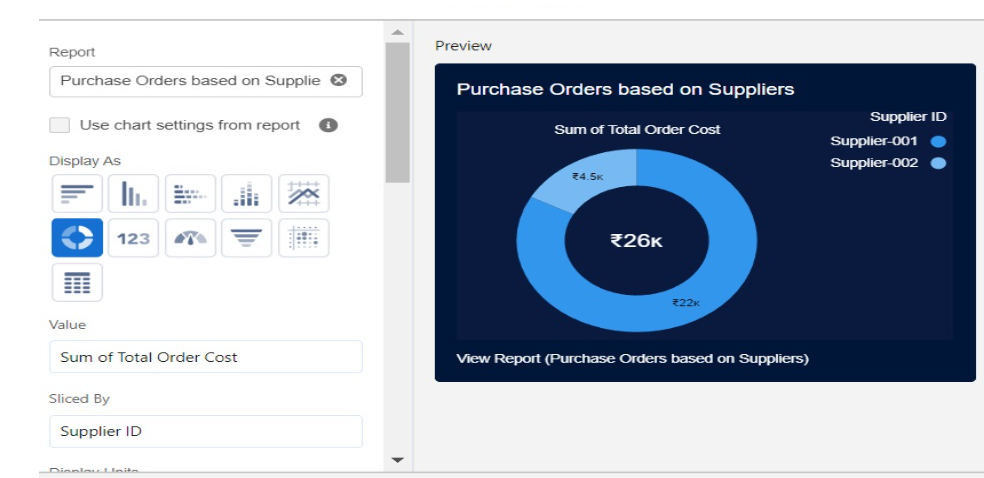
* Then to Create a Complete Purchase Details Report.
* Follow the same step to create a new report and name it as Complete Purchase Details Report.



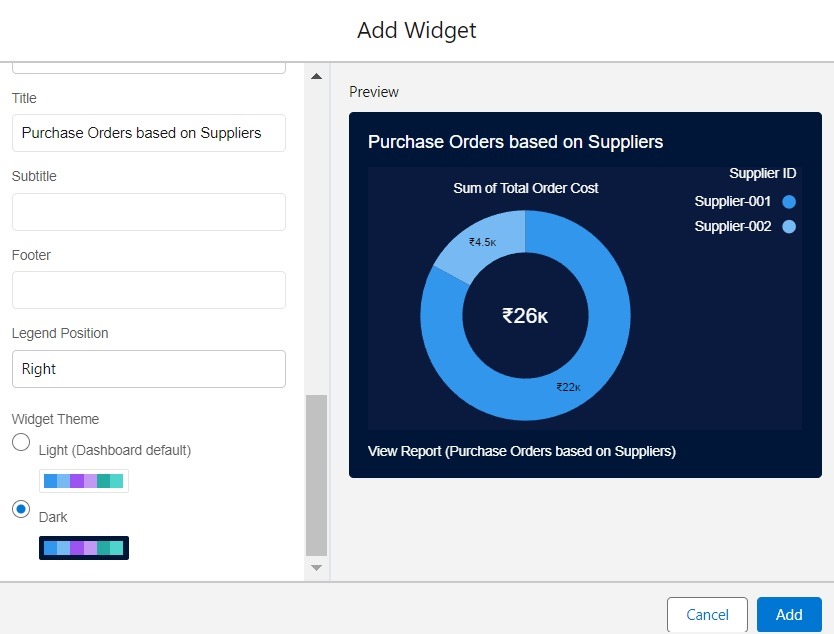
* Then save and run the report.
* The final step is to create and view a dashboard.



* Then select purchase order based on suppliers from reports.



* For the data visualization select any of the charts, tables etc. as per your choice/requirement.

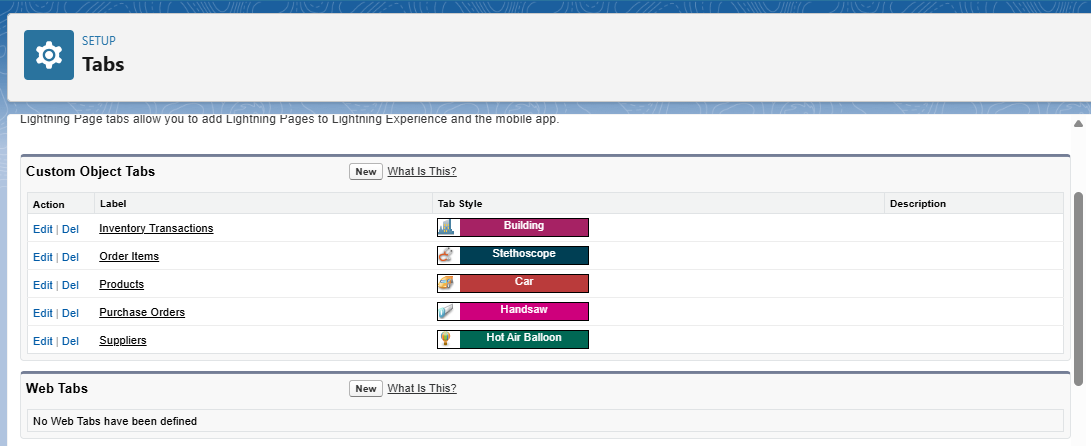


* To view the dashboard.Click on App Launcher on the left side of the screen.
* Search Medical Inventory Management & click on it.
* Click on Dashboard Tab.
* Click on Medical Inventory DashBoard see graph view of records.

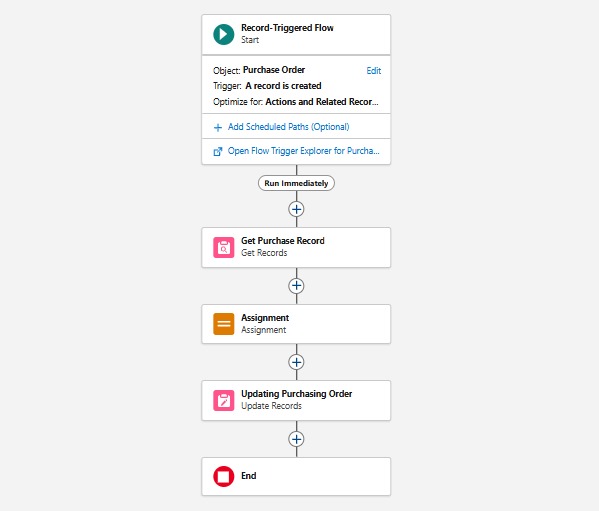
**RESULTS**

**Output Screenshots**

* Table for Product, Purchase order, Order item, Inventory Transactions, Supplier.

****

* Flow Runs



**CONCLUSION**

Medical Inventory Management is an essential system in the healthcare sector. By applying techniques like ABC, VED, FIFO, and modern technology such as RFID and automated software, hospitals can achieve efficient supply chain management. This ultimately leads to cost savings, reduced wastage, and better patient care.

**APPENDIX**

* **SOURCE CODE :** Provided in Apex Classes and Trigger

**CalculateTotalAmountTrigger.apxt :**

trigger CalculateTotalAmountTrigger on Order\_Item\_\_c (after insert, after update, after delete, after undelete) {

// Call the handler class to handle the logic

CalculateTotalAmountHandler.calculateTotal(Trigger.new, Trigger.old, Trigger.isInsert, Trigger.isUpdate, Trigger.isDelete, Trigger.isUndelete);

}

**CalculateTotalAmountHandler.apxc :**

public class CalculateTotalAmountHandler {

// Method to calculate the total amount for Purchase Orders based on related Order Items

public static void calculateTotal(List<Order\_Item\_\_c> newItems, List<Order\_Item\_\_c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete) {

// Collect Purchase Order IDs affected by changes in Order\_Item\_\_c records

Set<Id> parentIds = new Set<Id>();

// For insert, update, and undelete scenarios

if (isInsert || isUpdate || isUndelete) {

for (Order\_Item\_\_c ordItem : newItems) {

parentIds.add(ordItem.Purchase\_Order\_Id\_\_c);

}

}

// For update and delete scenarios

if (isUpdate || isDelete) {

for (Order\_Item\_\_c ordItem : oldItems) {

parentIds.add(ordItem.Purchase\_Order\_Id\_\_c);

}

}

// Calculate the total amounts for affected Purchase Orders

Map<Id, Decimal> purchaseToUpdateMap = new Map<Id, Decimal>();

if (!parentIds.isEmpty()) {

// Perform an aggregate query to sum the Amount\_\_c for each Purchase Order

List<AggregateResult> aggrList = [

SELECT Purchase\_Order\_Id\_\_c, SUM(Amount\_\_c) totalAmount

FROM Order\_Item\_\_c

WHERE Purchase\_Order\_Id\_\_c IN :parentIds

GROUP BY Purchase\_Order\_Id\_\_c

];

// Map the result to Purchase Order IDs

for (AggregateResult aggr : aggrList) {

Id purchaseOrderId = (Id)aggr.get('Purchase\_Order\_Id\_\_c');

Decimal totalAmount = (Decimal)aggr.get('totalAmount');

purchaseToUpdateMap.put(purchaseOrderId, totalAmount);

}

// Prepare Purchase Order records for update

List<Purchase\_Order\_\_c> purchaseToUpdate = new List<Purchase\_Order\_\_c>();

for (Id purchaseOrderId : purchaseToUpdateMap.keySet()) {

Purchase\_Order\_\_c purchaseOrder = new Purchase\_Order\_\_c(Id = purchaseOrderId, Total\_Order\_cost\_\_c = purchaseToUpdateMap.get(purchaseOrderId));

purchaseToUpdate.add(purchaseOrder);

}

// Update Purchase Orders if there are any changes

if (!purchaseToUpdate.isEmpty()) {

update purchaseToUpdate;

}

}

}

}